## CLAIMS

What is claimed:

1. A method of communicating between a plurality of processing nodes, the method comprising:

generating a channel that has bandwidth requirements and is uni-directional from a source processing node to a destination processing node;

accepting the channel in the destination processing node;

allocating a transmit buffer for the channel in the source processing node;

allocating a receive buffer for the channel in the destination processing node;

in a source processing element, writing data to the transmit buffer for the channel;

in a source network interface, transmitting the data from the transmit buffer of the source processing node over the channel;

in the destination network interface, receiving the data into a receive buffer for the channel in the destination processing node; and

in the destination processing element, receiving the data from the receive buffer.

2. The method of claim 1 wherein the channel is associated with a first task executing on the source processing element and a second task executing on the destination processing element.

3. The method of claim 1 wherein the channel is associated with a first port in the source processing element and a second port in the destination processing element.

4. The method of claim 1 wherein the channel has a maximum number and size of buffers.

5. The method of claim 1 further comprising reserving intermediate resources for the channel based on the bandwidth requirements.

6. The method of claim 1 further comprising guaranteeing bandwidth based on the bandwidth requirements using time division multiplexing.

7. The method of claim 1 further comprising guaranteeing bandwidth based on the bandwidth requirements using spatial division multiplexing.

8. The method of claim 1 further comprising polling a plurality of channels to check if data is received into the receive buffer for the channel.

9. The method of claim 1 further comprising freeing the transmit buffer.

10. The method of claim 1 further comprising freeing the receive buffer.

11. The method of claim 1 further comprising destroying the channel.

12. The method of claim 1 further comprising receiving a pointer for the data in the receive buffer into the destination processing element and wherein receiving the data from the receive buffer is based on the pointer.

13. The method of claim 1 wherein a time for a receive call in the destination processing element does not depend upon a size of the data.

14. A multi-processor system comprising:

a source processing node comprising:

a source processing element configured to generate a channel that has bandwidth requirements and is uni-directional to a destination processing node, allocate a transmit buffer for the channel, and write data to the transmit buffer for the channel; and

a source network interface configured to transmit the data from the transmit buffer of the source processing node over the channel; and

a destination processing node comprising:

a destination network interface configured to receive the data into a receive buffer for the channel; and

a destination processing element configured to accept the channel, allocate a receive buffer for the channel in the destination processing node, and receive the data from the receive buffer.

15. The multi-processor system of claim 14 wherein the channel is associated with a first task executing on the source processing element and a second task executing on the destination processing element.

16. The multi-processor system of claim 14 wherein the channel is associated with a first port in the source processing element and a second port in the destination processing element.

17. The multi-processor system of claim 14 wherein the channel has a maximum number and size of buffers.

18. The multi-processor system of claim 14 wherein the source processing node and the destination processing node are configured to reserve intermediate resources for the channel based on the bandwidth requirements.

19. The multi-processor system of claim 14 wherein the source processing node is configured to guarantee bandwidth based on the bandwidth requirements using time division multiplexing.

20. The multi-processor system of claim 14 wherein the source processing node is configured to guarantee bandwidth based on the bandwidth requirements using spatial division multiplexing.

21. The multi-processor system of claim 14 wherein the destination processing element is configured to poll a plurality of channels to check if data is received into the receive buffer for the channel.

22. The multi-processor system of claim 14 wherein the source processing element is configured to free the transmit buffer.

23. The multi-processor system of claim 14 wherein the destination processing element is configured to free the receive buffer.

24. The multi-processor system of claim 14 wherein the source processing element is configured to destroy the channel.

25. The multi-processor system of claim 14 wherein the destination processing element is configured to receive a pointer for the data in the receive buffer into the destination processing element and receive the data from the receive buffer based on the pointer.

26. The multi-processor system of claim 14 wherein a time for a receive call in the destination processing element does not depend upon a size of the data.

27. A method of compiling applications for a multi-processor system, the method comprising:

receiving a physical description of the multi-processor system;

receiving an application description indicating tasks for the applications and channels for communications between the tasks;

processing the physical description and the application description to determine routing information for the channels and to assign the tasks to processors in the multi-processor system; and

generating executable code for the processors based on the physical description and the application description.

28. The method of claim 27 wherein the physical description and the application description are in a package description.

29. The method of claim 27 wherein the physical description includes a configuration of processors in the multi-processor system.

30. The method of claim 27 wherein the application description includes application code for the tasks.

31. The method of claim 27 wherein the application description includes assignments of the tasks to execute on processors of the multi-processor system.

32. The method of claim 27 wherein the application description includes channels for communications.

33. The method of claim 32 wherein the application description includes routing for the channels.

34. The method of claim 27 wherein the application description includes shared memory descriptions.

35. The method of claim 27 further comprising processing the physical description and the application description to check for syntax and semantic errors.

36. The method of claim 27 wherein processing the physical description and the application description to determine routing information for the channels further comprises generating routing tables for the channels.

37. The method of claim 27 further comprising generating boot code for the processors in the multi-processor system.

38. The method of claim 27 further comprising setting scheduling policies for tasks executing in the processors.

39. The method of claim 27 wherein the executable code is for each processor based on a processor number.

40. The method of claim 27 further comprising mapping the executable code to memory in the multi-processor system.

41. The method of claim 27 wherein generating the executable code further comprises linking boot code, operating system code, and application code for the tasks.

42. The method of claim 27 wherein the executable code is executed on a host system for emulation.

43. The method of claim 27 wherein the executable code is executed on a simulator.

44. The method of claim 27 further comprising generating source code of the executable code for debugging.

45. The method of claim 27 further comprising:

determining assignment of the tasks in the multi-processor system based on the physical description;

determining the channels for communications between the tasks; and

generating the application description based on task assignments and the channels.

46. The method of claim 27 further comprising:

receiving performance data based on the execution of the executable code;

generating the application description based on the performance data.

47. A software product for compiling applications for a multi-processor system, the software product comprising:

package compiler software operational when executed by a first processor to direct the first processor to receive a physical description of the multi-processor system, receive an application description indicating tasks for the applications and channels for communications between the tasks, process the physical description and the application description to determine routing information for the channels and to assign the tasks to second processors in the multi-processor system, and generate executable code for the second processors based on the physical description and the application description; and

a software storage medium operational to store the package compiler software

48. The software product of claim 47 wherein the physical description and the application description are in a package description.

49. The software product of claim 47 wherein the physical description includes a configuration of the second processors in the multi-processor system.

50. The software product of claim 47 wherein the application description includes application code for the tasks.

51. The software product of claim 47 wherein the application description includes assignments of the tasks to execute on the second processors of the multi-processor system.

52. The software product of claim 47 wherein the application description includes channels for communications.

53. The software product of claim 52 wherein the application description includes routing for the channels.

54. The software product of claim 47 wherein the application description includes shared memory descriptions.

55. The software product of claim 47 wherein the package compiler software is operational when executed by the first processor to direct the first processor to process the physical description and the application description to check for syntax and semantic errors.

56. The software product of claim 47 wherein the package compiler software is operational when executed by the first processor to direct the first processor to generate routing tables for the channels.

57. The software product of claim 47 wherein the package compiler software is operational when executed by the first processor to direct the first processor to generate boot code for the second processors in the multi-processor system.

58. The software product of claim 47 wherein the package compiler software is operational when executed by the first processor to direct the first processor to set scheduling policies for tasks executing in the second processors.

59. The software product of claim 47 wherein the executable code is for each second processor based on a processor number.

60. The software product of claim 47 wherein the package compiler software is operational when executed by the first processor to direct the first processor to map the executable code to memory in the multi-processor system.

61. The software product of claim 47 wherein the package compiler software is operational when executed by the first processor to direct the first processor to link boot code, operating system code, and application code for the tasks to generate the executable code.

62. The software product of claim 47 wherein the executable code is executed on a host system for emulation or simulation.

63. The software product of claim 47 wherein the executable code is executed on a simulator.

64. The software product of claim 47 wherein the package compiler software is operational when executed by the first processor to direct the first processor to generate source code of the executable code for debugging.

65. The software product of claim 47 wherein the package compiler software is operational when executed by the first processor to direct the first processor to determine assignment of the tasks in the multi-processor system based on the physical description, determine the channels for

communications between the tasks, and generate the application description based on task assignments and the channels.

66. The software product of claim 47 wherein the package compiler software is operational when executed by the first processor to direct the first processor to receive performance data based on the execution of the executable code and generate the application description based on the performance data.

67. A method of booting a multi-processor system comprising a root processor and at least one non-root processor, the method comprising:

identifying the root processor in the multi-processor system that does not have memory associated with the at least one non-root processor;

transmitting a boot message from the root processor to the at least one non-root processors;

receiving the boot message into the at least one non-root processor;

in the at least one non-root processor, obtaining the non-root boot code based on the message in the non-root code; and

configuring the at least one non-root processor based on the non-root boot code.

68. The method of claim 67 further comprising executing processor initialization code for initialization of the root processor.

69. The method of claim 67 further comprising executing processor initialization code for initialization of the at least one non-root processor.

70. The method of claim 67 wherein the boot message is a Joint Test Action Group command.

71. The method of claim 67 wherein transmitting the boot message is from a Joint Test Action Group port.

72. The method of claim 67 wherein receiving the boot message is into a Joint Test Action Group port.

73. The method of claim 67 further comprising:

in the at least one non-root processor, transmitting a boot complete message to the root processor;

in the root processor, transmitting a proceed message in response to the boot complete message; and

in the at least one non-root processor, beginning operations in response to the proceed message.

74. A multi-processor system comprising:

a root processor configured to identify the root processor as a root and transmit a boot message from the root processor to the at least one non-root processors;

the at least one non-root processor that does not have memory associated with the at least one non-root processor and that is configured to receive the boot message, obtain the non-root boot code based on the message in the non-root code, and configuring the at least one non-root processor based on the non-root boot code.

75. The multi-processor system of claim 74 wherein the root processor is configured to execute processor initialization code for initialization of the root processor.

76. The multi-processor system of claim 74 wherein the at least one non-root processor is configured to execute processor initialization code for initialization of the at least one non-root processor.

77. The multi-processor system of claim 74 wherein the boot message is a Joint Test Action Group command.

78. The multi-processor system of claim 74 wherein the root processor is configured to transmit the boot message from a Joint Test Action Group port.

79. The multi-processor system of claim 74 wherein the at least one non-root processor is configured to receive the boot message into a Joint Test Action Group port.

80. The multi-processor system of claim 74 wherein the at least one non-root processor is configured to transmit a boot complete message to the root processor and begin operations in response to a proceed message and wherein the root processor is configured to transmit the proceed message in response to the boot complete message.